Popular Computing

76

The world's only magazine devoted to the art of computing.

July 1979 Volume 7 Number 7

400	399	398	397	396	395	394	393	392	391	390	389	388	387	386	385	384	383	382	381
325	324	323	322	321	320	319	318	317	316	315	314	313	312	311	310	309	308	307	380
326	257	256	255	254	253	252	251	250	249	248	247	246	245	244	243	242	241	306	379
327	258	197	196	195	194	193	192	191	190	189	188	187	186	185	184	183	240	305	378
328	259	198	145	144	143	142	141	140	139	138	137	136	135	134	133	182	239	304	377
329	260	199	146	101	100	99	98	97	96	95	94	93	92	91	132	181	238	303	376
330	261	200	147	102	65	64	63	62	61	60	59	58	57	90	131	180	237	302	375
331	262	201	148	103	66	37	36	35	34	33	32	31	56	89	130	179	236	301	374
332	263	202	149	104	67	38	17	16	15	14	13	30	55	88	129	178	235	300	373
333	264	203	150	105	68	39	18	5	4	3	12	29	54	87	128	177	234	299	372
334	265	204	151	106	69	40	19	6	1	2	11	28	53	86	127	176	233	298	371
335	266	205	152	107	70	41	20	7	8	9	10	27	52	85	126	175	232	297	370
336	267	206	153	108	71	42	21	22	23	24	25	26	51	84	125	174	231	296	369
337	268	207	154	109	72	43	44	45	46	47	48	49	50	83	124	173	230	295	368
338	269	208	155	110	73	74	75	76	77	78	79	80	81	82	123	172	229	294	367
339	270	209	156	111	112	113	114	115	116	117	118	119	120	121	122	171	228	293	366
340	271	210	157	158	159	160	161	162	163	164	165	166	167	168	169	170	227	292	365
341	272	211	212	213	214	215	216	217	218	219	220	221	222	223	224	225	226	291	364
342	273	274	275	276	277	278	279	280	281	282	283	284	285	286	287	288	289	290	363
343	344	345	346	347	348	349	350	351	352	353	354	355	356	357	358	359	360	361	362

20 x 20

20 x 20

On the front cover of this issue is a 20 x 20 array of the numbers from 1 to 400 arranged in a spiral. The cells in the array are identified by columns, from one to 20, left to right, and by rows, from one to 20, top to bottom. Each cell can thus be labelled as A(I,J). Thus we wish to have:

A(1,1) = 400
A(2,2) = 324
A(3,3) = 256
. .
. .
A(18,18) = 226
A(19,19) = 290
A(20,20) = 362

...examples...

Part 1 of this Problem, then, is to generate the 400 numbers in the proper cells in the array A.

We then wish to be able to shift any row or column by any number of cells, in circular fashion. That is, we want a subroutine to shift row Q, R cells to the right, with the numbers shifted off the array at the right appearing in the same row on the left end. Thus, for Q = 11, R = 3, row 11 will become:

233 298 371 334 265 .... 53 86 127 176

If R = 20, there will be much activity, but the row will appear unchanged. To shift a row to the left by W places, let R = (20 - W).

Similarly, we want a subroutine that will shift column S up by T cells. Again, if T = 20, the column will appear unchanged. To shift a column of the array down by Z, let T = (20 - Z).

☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆☆

Publisher: **Audrey Gruenberger**
Editor: **Fred Gruenberger**
Associate Editors: **David Babcock**
Irwin Greenwald
Patrick Hall
Contributing Editors: **Richard Andree**
William C. McGee
Thomas R. Parkin
Edward Ryan
Art Director: **John G. Scott**
Business Manager: **Ben Moore**

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year. For all other countries, add $3.50 per year. Back issues $2.50 each. Copyright 1979 by POPULAR COMPUTING.

The center of the array is this:

	Col. 10	Col. 11
Row 10	4	3
Row 11	1	2

The following actions are to be performed:

Shift	row	11	1 place to the right
Shift	column	11	1 place up
Shift	row	10	1 place to the left
Shift	column	10	1 place down.

Following that, the rows and columns just outside the center (namely, rows 9 and 12; columns 9 and 12) are to be similarly shifted <u>two</u> places; that is,

Shift	row	12	2 places to the right
Shift	column	12	2 places up
Shift	row	9	2 places to the left
Shift	column	9	2 places down.

This procedure continues, moving out from the center, until finally we:

Shift	row	20	10 places to the right
Shift	column	20	10 places up
Shift	row	1	10 places to the left
Shift	column	1	10 places down.

The final status of the array is shown on the back cover.

Producing that result is Part 2 of this Problem.

Everything called for is now done, and the result is recorded. Precisely; and we now have a splendid coding assignment, readily defined, with an enormous amount of detailed work to be performed--all leading to a known result (that is, it is known if <u>we</u> have done our work correctly). Try it--it appears to be quite simple.

Take/Skip Revisited

Probably the most spectacular problem we have so far published (and certainly the one that attracted the most interest and work) first appeared as the K-Level Sieve in issue 38, but was soon dubbed the Take/Skip Problem. A brilliant solution by two University of Toronto students, Tom Duff and Hugh Redelmeier, appeared in issue 43.

Contributing Editor Edward Ryan observed that the technique used in "...Or Not Recurse" in issue 75 could apply to the Take/Skip problem, and that therefore a fairly efficient implementation of Duff and Redelmeier's solution could be made in elementary BASIC. Let us restate the original problem:

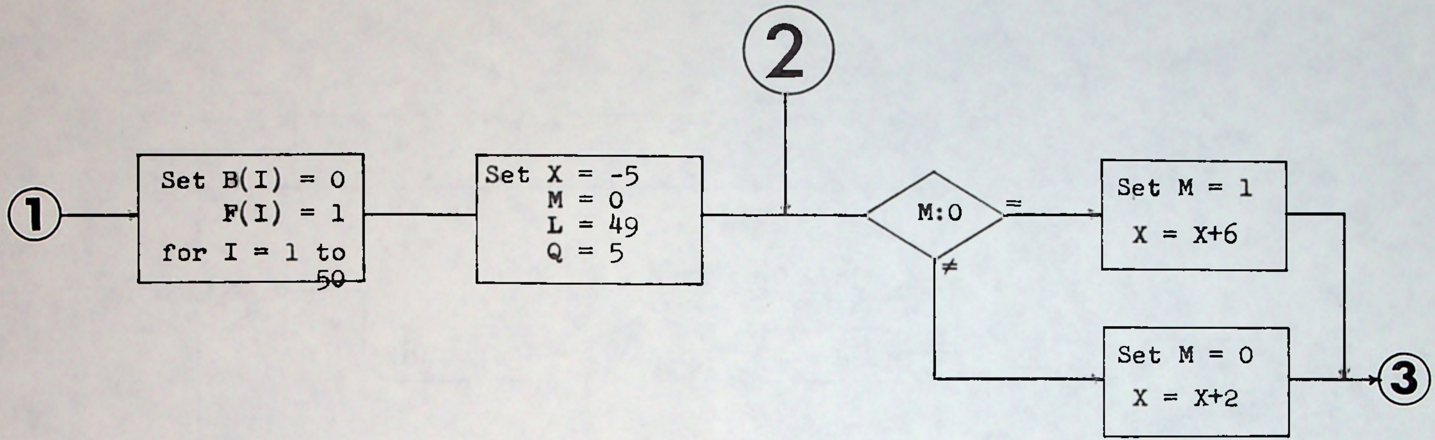
> Start with the positive integers. At each level, K, of the procedure to be followed, the numbers that are outputted from level K-1 are to be sieved by accepting K numbers and rejecting K numbers, alternately. What numbers will survive all levels of sieving?
>
> The positive integers form level zero. Level 1 accepts one number and rejects the next; thus the output of level 1 (which constitutes the input to level 2) is the sequence of odd numbers.

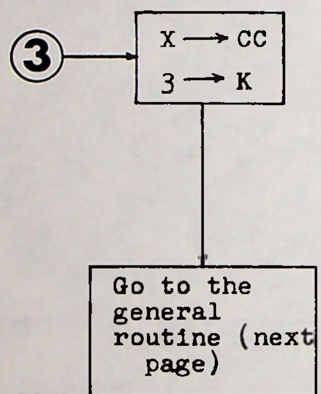
Subsequent levels will deal with the following sequences:

2	1	3	9	11	17	19	25	27	33	35	41	43
3	1	3	9	25	27	33	49	51	57	73	75	81
4	1	3	9	25	57	73	75	81	123	129	145	147
5	1	3	9	25	57	145	147	193	195	201	...	

Each level passes on to the next level exactly half of the numbers it receives. Thus, for a million numbers treated at level zero, just one number will emerge at level 19.

The point to focus on, however, is that each level treats just one number at a time, and rejects it (and that number is then seen no more) or accepts it and passes it on to the next level. So if we observe the numbers that survive to any given level, say level L, then as long as those numbers are less than 2 to the power L, they are numbers that will survive all levels of the sieve.





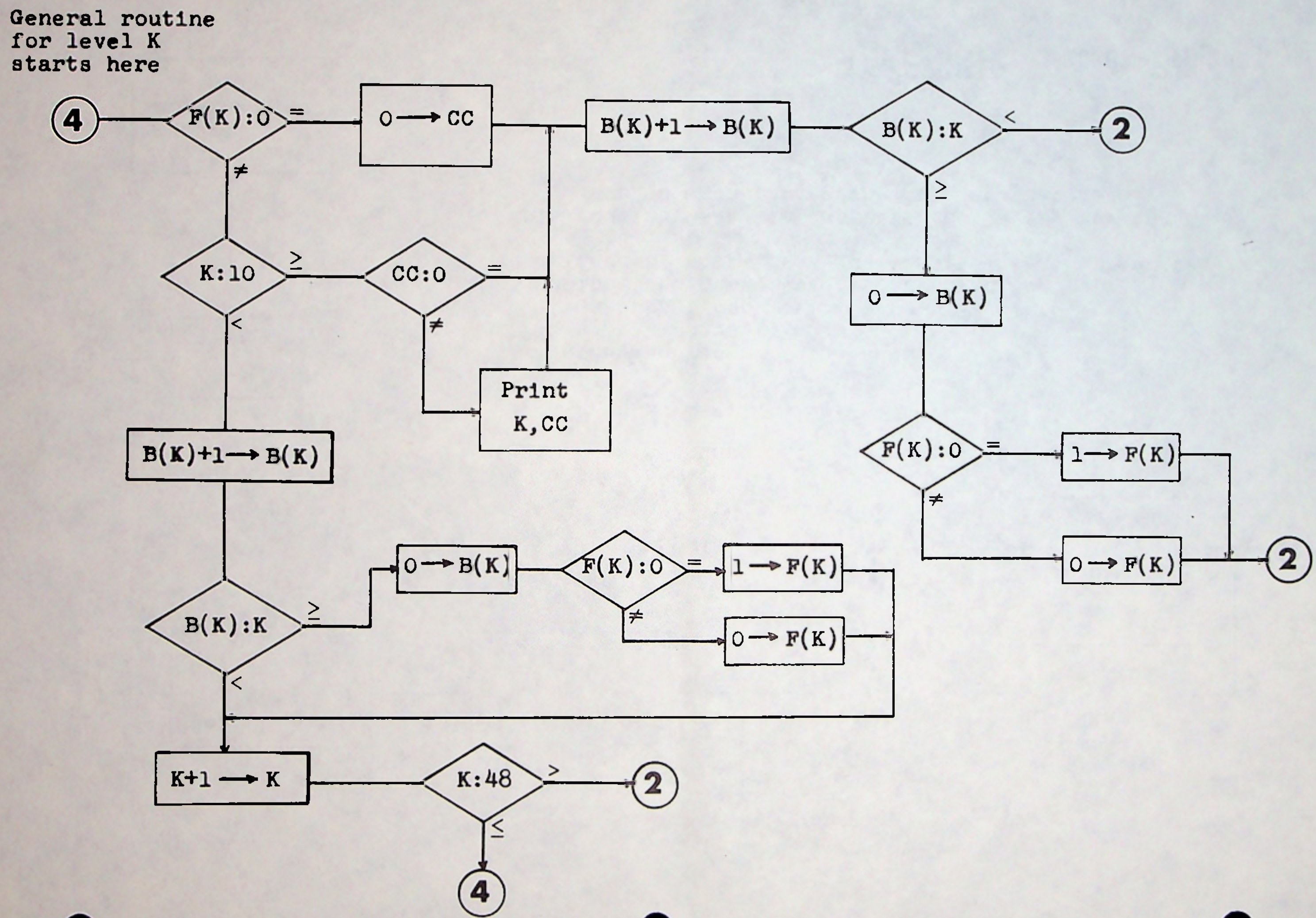
K is the level number
Each B is a counter for the corresponding level K

Each F is a flag, at level K: 0 = reject; 1 = accept

X is the generator of the data
M is a switch control
CC is a bucket for moving X from level to level.
L defines the number of levels to go to
Q defines at what level you wish to see the results

This routine, starting at reference 2, is working at level 2, generating the data at that level.

TAKE/SKIP Bucket Brigade


General routine
for level K
starts here
4
F(K):0
=
≠
0 ⟶ CC
B(K)+1 ⟶ B(K)
B(K):K
<
≥
2
K:10
≥
<
CC:0
=
≠
Print
K,CC
0 ⟶ B(K)
F(K):0
=
≠
1 ⟶ F(K)
0 ⟶ F(K)
2
B(K)+1 ⟶ B(K)
B(K):K
≥
<
0 ⟶ B(K)
F(K):0
=
≠
1 ⟶ F(K)
0 ⟶ F(K)
K+1 ⟶ K
K:48
>
≤
2
4

So all we need is a bunch of counters and flags. The flowcharts given here have been made to operate through level 50 (and 1125899906842624 is the 50th power of 2).

We set up a counter, B, for each level, and all the B's are initialized to zero. We also establish a flag, F, at each level, all initialized to one (standing for "accept"). The mechanism at Reference 2 generates the level 2 output. Beginning with level 3 (K = 3), all levels operate the same way, as shown in the second flowchart.

Consider level 10. It receives numbers, one at a time, from level 9 (transmitted via word CC). Starting from the beginning, it should pass 10 numbers on to level 11, then reject the next 10 numbers it receives, then pass on the next 10, and so on. Its counter, B(10), counts up to the level number 10, and then resets to zero to get ready to count to 10 again. Its flag, F(10), flipflops from zero to one to zero to indicate whether it is in the accept (1) or reject (0) mode. When any level is in reject mode, the contents of CC is made zero, and control returns to Reference 2, to generate the next number at level 2.

Even in interpretive BASIC, this procedure is reasonably fast. A possible program is included.

☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆ ☆

Most Back Issues Are Still Available:

JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC		
			1	2	3	4	5	~~6~~	~~7~~	~~8~~	9	Vol. 1	1973
10	11	12	~~13~~	14	~~15~~	~~16~~	17	~~18~~	19	~~20~~	21	Vol. 2	1974
22	23	24	25	26	27	28	229	30	31	32	33	Vol. 3	1975
34	~~35~~	36	37	38	39	40	41	42	43	44	45	Vol. 4	1976
46	47	48	49	50	51	52	53	54	55	56	57	Vol. 5	1977
58	59	60	61	62	63	64	65	66	67	68	69	Vol. 6	1978
70	71	72	73	74	75							Vol. 7	1979

Possible program in BASIC for the TAKE/SKIP Bucket Brigade

Line 500 is Reference 2 of the flowchart

Line 700 is Reference 3 of the flowchart

```
100 DIM B(50)
110 DIM F(50)
120 FØR I = 1 TØ 50
130 B(I) = 0
140 F(I) = 1
150 NEXT I
160 X = -5
170 M = 0

500 IF M = 0 THEN 600
510 X = X + 2
520 M = 0
530 GØ TØ 700

600 M = 1
610 X = X + 6

700 CC = X
710 K = 3

1000 IF F(K) = 0 THEN 1100
1010 IF K > 10 THEN 1200
1020 B(K) = B(K) + 1
1030 IF B(K)>=K THEN 1300
1040 B(K) = 0
1050 IF F(K) = 0 THEN 1400
1060 F(K) = 0
1070 GØ TØ 1300

1100 CC = 0
1110 B(K) = B(K) + 1
1120 IF B(K) < K THEN 500
1130 B(K) = 0
1140 IF F(K) = 0 THEN 1500
1150 F(K) = 0
1160 GØ TØ 500

1200 IF CC = 0 THEN 1020
1210 PRINT K, CC
1220 GØ TØ 1020

1300 K = K + 1
1310 IF K > 48 THEN 500
1320 GØ TØ 1000

1400 F(K) = 1
1410 GØ TØ 1300

1500 F(K) = 1
1510 GØ TØ 500
```

Penny Flipping VII

There are two stacks of coins numbering C coins in each stack. Initially, both stacks are all heads up. The accompanying illustration is for C = 3.

The first stack follows the procedure of the first Penny Flipping problem; namely:

> Flip the top coin, then the top two, then the top three,..., and so on until the entire stack is flipped; then start over with the top one, the top two,...

The number of coins that are tails up in the first stack at each stage is the number to be flipped in the second stack. Thus, the top coin of the first stack is flipped to start, which gives one tail, which dictates flipping one coin of the second stack. This procedure continues until both stack again become all heads up. For C = 3, this takes 36 flips, as shown. The results for cases 2 through 40 are given.

The results for previous penny flipping problems were distributions that oscillated wildly. This latest version, due to Ralph Montgomery of St. John's Community College, which would appear to be even wilder from its description, turns out to be (at least for 40 cases) remarkably well behaved.

As usual, the point of the problem is to provide an interesting coding exercise (the penny flipping problems lend themselves well to integer BASIC or machine language) for which test results are available. We solicit further results for values of C greater than 40.

```
0 1 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0 1 1 1 0 0 1 0 1 0
0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 0
0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 1 1 1 0

0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 1 0 1 1 0 1 1 0 0 0 1 0 0 1 0 0 1 0 0
0 0 0 1 1 1 0 0 1 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
```

2	3
3	36
4	48
5	100
6	288
7	112
8	256
9	972
10	1800
11	3630
12	2640
13	13104
14	14112
15	1500
16	26880
17	4896
18	7776
19	5928
20	22400
21	10584
22	166320
23	63480
24	44352
25	12000
26	1784640
27	83160
28	332640
29	90828
30	187200
31	34224
32	59904
33	670824
34	125664
35	465500
36	221616
37	229400
38	307800
39	267696
40	8078400

The distribution of results is fairly smooth, except for the two anomalous numbers here.

More results, April 1979

41	12647844
42	766080
43	433440
44	107448
45	32400
46	154560
47	145512
48	193536
49	229320
50	13110000

Penny flipping VII results March 1979

THROWBACK Revisited

The THROWBACK problem (Number 193) was on the cover of issue number 55. The problem was this:

Given all the positive integers starting with 3:

3 4 5 6 7 8 9 10 11 12 13 14

The leading 3 specifies throwing it (the 3) back 3 places, putting it between the 6 and 7. This process now repeats, as shown here:

4 5 6 3 7 8 9 10 11 12 13 14

5 6 3 7 4 8 9 10 11 12 13 14

6 3 7 4 8 5 9 10 11 12 13 14

3 7 4 8 5 9 6 10 11 12 13 14

and we count the number of throws to bring each higher number to the leading position, giving a table:

4	1
5	2
6	3
7	5
8	7
9	10
10	14
11	19

In issue 58, this table was extended to line 48, largely as a result of work done by William Bourn, who did it with a program in APL. The current interest in BASIC prompts us to present a program in that language to implement a solution. But this leads us to a new problem.

In the original version, the results <u>appear</u> to follow the rule:

$$F_{n+1} = \left[4/3 \cdot F_n + 1 \right],$$

at least for the first 48 results. This indicates that the number 100 will first appear at the head of the stream of numbers after some 2,731,288,000,000 throws have been made.

Now suppose that the starting sequence of numbers began with 10, rather than 3. In the BASIC program, these two changes will get the result empirically:

```
120 A(I) = I + 9

140 X = 11
```

and the resulting table begins:

11	1	19	9	27	26
12	2	20	10	28	29
13	3	21	12	29	32
14	4	22	14	30	36
15	5	23	16	31	40
16	6	24	18	32	45
17	7	25	20	33	50
18	8	26	23	34	56

The new problem is to reach a logical conclusion as to when the number 100 will first arrive at the head of the stream.

```
100 DIM A(100)
110 FØR I = 1 TØ 100
120 A(I) = I + 2
130 NEXT I
140 X = 4
150 N = 0

200 K = A(1)
210 FØR I = 1 TØ K
220 A(I) = A(I+1)
230 NEXT I
240 A(K+1) = K
250 N = N + 1
260 IF A(1) = X THEN 500
270 GØTØ 200

500 PRINT X, N
510 X = X + 1
520 GØTØ 200
530 END
```

Program in BASIC for the original THROWBACK problem.

Penny Flipping Again

Hopefully, a final report on the first Penny Flipping Problem. See previous reports in issues 23, 25, 71, and 73.

The last reference (issue 73, page 18) reported on research by David E. Ferguson, he who runs the Ferguson Tool Company. Mr. Ferguson adds the following results now:

$2^k \pmod a$ can be computed in at most 2k adds (average 3k/2 adds). Since $\Phi(2C+1)$ is always even,

$$2^{\Phi(2C+1)/2} \equiv \pm 1 \pmod{2C+1}$$

and since $\Phi(2C+1) \le 2C$, $n \le C$, n can be computed in at most 2C adds (average 3C/4 adds).

Also, if 2C+1 is prime and

(a) $C \equiv 1$ or $C \equiv 2 \pmod 4$ $f(C) = C^2/d-1$ where d is a proper odd divisor of C.

Corollary 1: If C is a prime $\equiv 1 \pmod 4$, then $f(6) = C^2 - 1$

Corollary 2: If C is twice a prime, $f(C) = C^2-1$

(b) $C \equiv 3 \pmod 4$ $f(C) = C^2/d$ where d is a proper divisor of C.

Corollary 3: If C is a prime $\equiv 3 \pmod 4$, then $f(C) = C^2$

(c) $C \equiv 0 \pmod 4$, either

(C1) $f(C) = C^2/d-1$ where d is a proper even divisor of C; or

(C2) $f(C) = Cd$ where d is a proper odd divisor of C.

(d) $f(C)$ is of the form $nC-1$ if and only if $f(p_1)$ is of the form n_1p_1-1 for every prime divisor, p_1, of $2C+1$. (This accounts for the thinning out of this form for higher values of C.)

(e) A general expression for $f(C)$ can be written in terms of the prime factorization of $2C+1$ which will work in all but very rare cases; namely, when $2^{p-1} \equiv 1 \pmod{p^2}$; e.g., when $p = 1093$.

(f) My "f(C)" is your "N"; my "n" in $2^n \equiv \pm 1$ is not related to your "N".

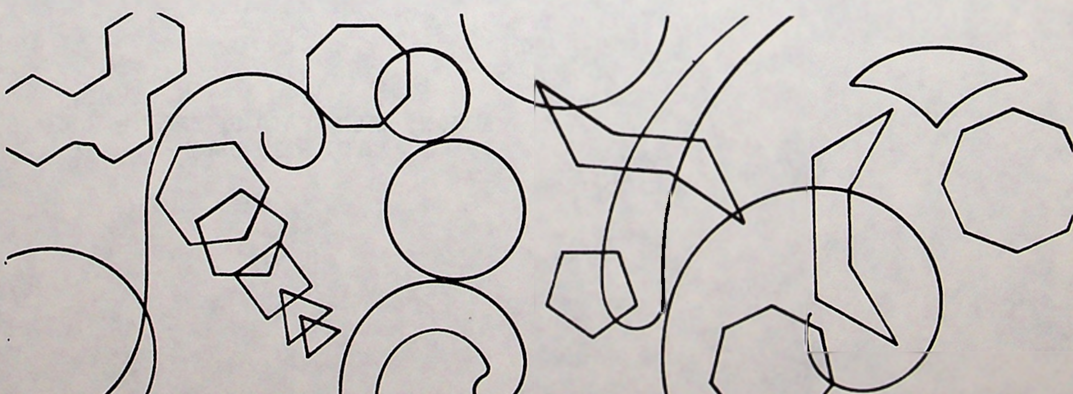
Meanwhile, using very little analytics, but clever coding (by Associate Editor David Babcock) and enormous amounts of computing power, the table of results for the first penny flipping problem from issue 71 can now be extended. For stacks of coins from 240 to 539, and from 1000 to 1103, the number of flips is given in the accompanying table.

In the review of the TRS-80 by Larry Clark in issue 75, Mr. Clark was credited with appearing in the film "JOSS." The confusion was natural, inasmuch as he was in charge of the JOSS project at The RAND Corporation for some time. The film appearance, however, was in the AFIPS film "It's Your Move."

240	4319	290	71340	340	38419
	15906		75660		3750
	11616		3504		11627
	59049		85848		26068
	19763		26460		26831
	60024		57820		39674
	13776		21903		10380
	14820		7128		95772
	26040		59004		13920
	20666		89401		20242
250	41500	300	7500	350	122499
	63001		9932		12636
	12599		66440		32384
	39468		91809		105900
	64515		25536		125315
	2295		84180		27690
	2303		93635		19580
	52428		6140		21420
	59340		23715		85204
	44548		95480		128881
260	33799	310	61380	360	18360
	68120		10263		8664
	15720		77999		25339
	10520		14084		43923
	66792		22608		88451
	23054		14175		20440
	7979		33179		44651
	56604		8876		30828
	23851		26712		60719
	56490		66990		45386
270	72899	320	10239	370	13320
	48780		34346		137641
	4895		9016		27527
	74528		104329		45878
	16440		46979		118932
	69300		9750		140625
	10764		106275		9399
	9972		85020		22620
	77283		5904		142883
	23436		108240		41690
280	11200	330	108899	380	72199
	78960		7944		13716
	3947		11952		9168
	15282		102564		133284
	40327		24716		73727
	16244		20100		6160
	54340		8063		148995
	63140		60660		7740
	20735		114243		13968
	27744		16272		70020

390	27300	440	24200	490	17640
	98532		194480		241081
	10191		51272		48215
	154448		196249		68034
	103228		9324		76076
	33180		60074		245025
	11879		184644		7439
	20644		159132		196812
	158403		59136		82667
	73416		62860		17964
400	26400	450	46800	500	30000
	36090		18942		116232
	53064		40679		66264
	108004		205208		235404
	81607		136200		127007
	54674		41405		21210
	109620		93479		23275
	131868		27420		42588
	25703		178620		42672
	4908		70227		259080
410	168099	460	23459	510	86699
	168921		193620		5110
	8240		41579		5119
	170568		47226		80028
	171395		107647		151116
	38180		58590		265225
	69888		144460		66563
	138444		18680		68244
	37620		54756		62160
	175561		73164		269361
420	170519	470	220899	520	89959
	14734		103620		231324
	32915		16992		93960
	139590		223728		182004
	19927		17064		68643
	168300		150100		91874
	181475		16183		56808
	15372		181260		221340
	91591		66920		7920
	184040		97716		46552
430	25800	480	74400	530	280899
	185761		76478		281961
	37151		46272		74480
	58888		233289		127920
	169260		34848		95051
	57420		47044		12840
	20928		67068		67535
	131100		29220		75180
	191843		119071		192604
	128188		53790		265188

1000	308000	1040	540799	1080	583199
	143142		1083680		110262
	200400		287592		77904
	222666		1087849		1061340
	421680		30276		26016
	202004		41800		1080660
	60360		138072		141179
	60420		875292		152180
	169343		182352		505920
	48432		1100400		395306
1010	325220	1050	199500	1090	263780
	412488		735700		1139004
	546480		220919		432432
	1026168		44226		796796
	1028195		37944		1083060
	686140		1113025		170820
	484631		23231		61376
	183060		291732		320324
	48864		266616		1113371
	1038361		343116		268156
1020	79559	1060	318000	1100	38500
	346118		509280		404066
	104243		212400		92568
	11253		752604		1216609
	11263		283023		
	897900		1134224		
	1052675		249444		
	69836		64020		
	452320		570311		
	144060		117590		
1030	234840	1070	1144899		
	1062961		54621		
	359136		64320		
	161148		270396		
	1069155		109548		
	37260		767550		
	119139		578887		
	850340		185244		
	342540		774004		
	93510		60424		



Palindromic Numbers

As part of a larger problem, it is important to know whether or not the decimal representation of a number is a palindrome. A palindrome is something which "reads" the same forwards and backwards.

For example, the numbers:

1 44 323 5775 and -848

are palindromes, while the numbers:

12 344 67 -378 and 5050

are not.

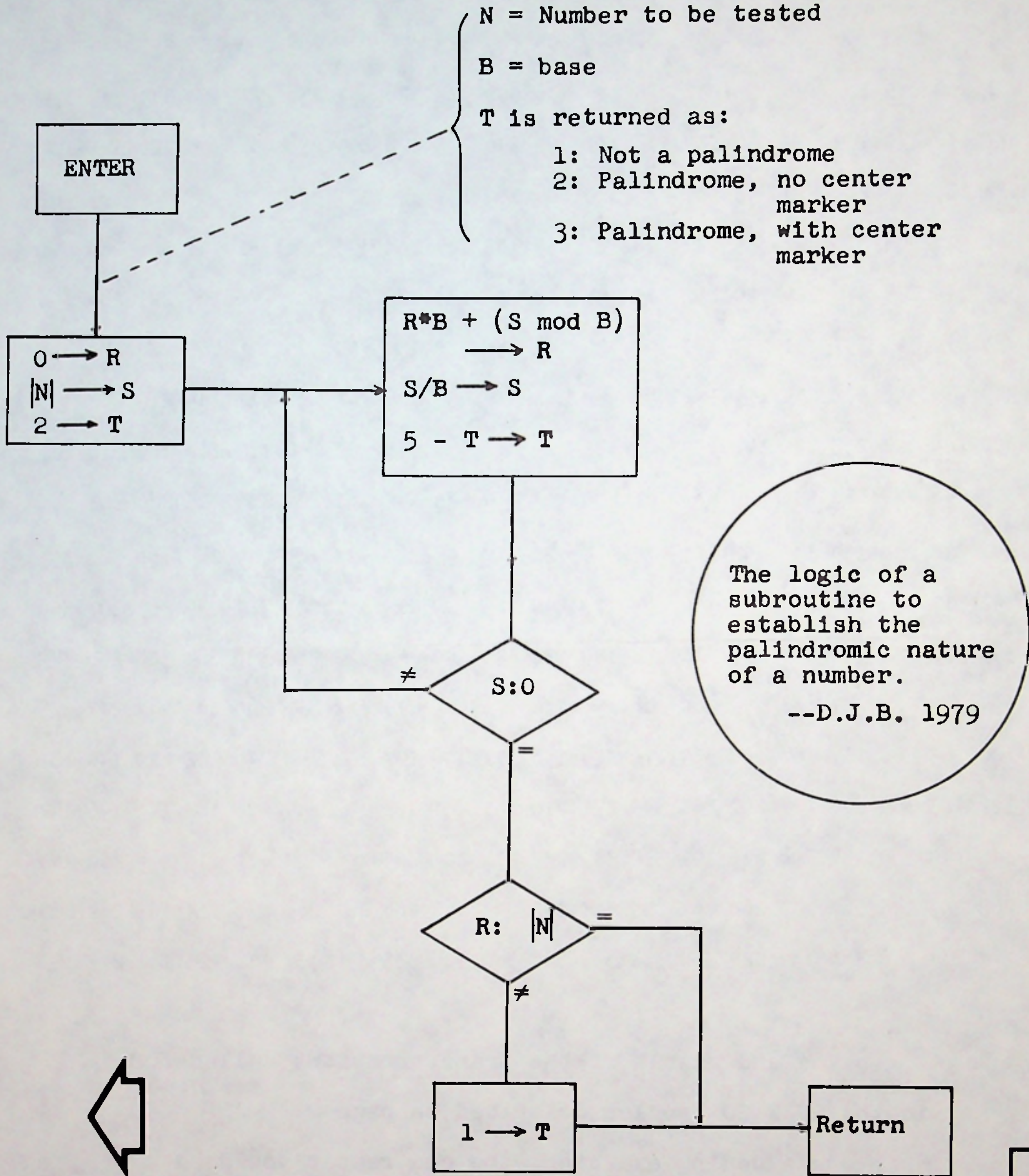
The program segment below and the accompanying flowchart perform the required task.

```
INTEGER PRØCEDURE PALINDRØME (N,B);
      INTEGER N,B;

BEGIN INTEGER R,S,T;
      R:=0;
      S:=ABS(N);
      T:=2;

   DØ BEGIN
      R:= R*B + MØD(S,B);
      S:= S/B;
      T:= 5 - T;
      END
   UNTIL S = 0;

   IF R<>ABS(N) THEN PALINDRØME := 1
                ELSE PALINDROME := T;
END;
```



The logic of a subroutine to establish the palindromic nature of a number.

--D.J.B. 1979

371	247	187	135	91	375	374	373	372	298	400	370	369	368	367	112	160	216	280	352
297	296	188	136	92	302	301	300	299	233	380	325	324	295	294	293	159	215	279	232
231	230	229	137	93	57	236	235	234	176	306	379	326	257	256	228	158	214	175	174
173	172	171	170	94	58	179	178	177	127	240	305	378	327	258	197	196	126	125	124
123	122	243	212	276	59	31	129	128	86	182	239	304	377	328	259	85	84	83	82
386	311	244	275	347	395	32	88	87	53	132	181	238	303	376	52	51	56	285	356
387	312	274	346	396	320	252	13	54	28	90	131	180	237	27	26	166	221	284	355
388	313	345	397	321	253	193	141	29	11	56	89	130	10	79	118	165	220	283	354
389	344	398	322	254	194	142	98	62	2	30	55	24	47	78	117	164	219	282	353
390	399	323	255	195	143	99	63	35	15	12	6	8	23	46	77	116	163	218	281
315	248	189	138	95	60	33	14	3	1	9	25	49	81	121	169	225	289	361	381
391	316	249	190	139	96	61	34	4	20	7	48	80	120	168	224	288	360	382	351
392	317	250	191	140	97	16	5	71	42	21	19	119	167	223	287	359	383	307	350
393	318	251	192	36	17	18	154	109	72	43	40	41	222	286	358	384	308	278	349
394	319	64	37	38	39	269	208	155	110	73	69	70	22	357	385	309	241	277	348
100	65	66	67	68	366	339	270	209	156	111	106	107	108	45	310	242	213	144	101
102	103	104	105	227	292	365	340	271	210	157	151	152	153	44	76	183	145	146	147
148	149	150	184	198	226	291	364	341	272	211	204	205	206	207	75	115	199	200	201
202	203	245	185	133	260	261	290	363	342	273	265	266	267	268	74	114	162	262	263
264	314	246	186	134	329	330	331	332	362	343	334	335	336	337	338	113	161	217	333

The end result of the transformations called for in the 20 x 20 Problem described on page 2.

As a coding exercise, the end result would be checked sufficiently by printing just the main diagonal elements of the array:

A(1,1) = 371
A(2,2) = 296
. . .   . .
A(20,20) = 333